**MIDTERM SUBMISSION**

SOC:41

Facial recognition

Shravanee Kulkarni

24b2206

# **BASICS of Python**

## **Input and Output**

Python[**input() function**](https://www.geeksforgeeks.org/python-3-input-function/) is used to take user input. By default, it returns the user input in form of a string

name = input("Enter your name: ")

or

name = input(‘Enter your name: ‘)

Python takes input **both in single and double quotes**

The output: print("Hello, World!")

## **Variables in Python**

variables are used to store data that can be referenced and manipulated during program execution. Python variables do not require explicit declaration of type. The type of the variable is inferred based on the value assigned.

# Variable 'x' stores the integer value 10  
x = 5  
# Variable 'name' stores the string "blah"  
name = "blah"

# Define variables with different data types  
n = 42 //int

f = 3.14 //float

s = "Hello, World!" //string

li = [1, 2, 3] //list

d = {'key': 'value'} //dict

bool = True //bool

## **Rules for Naming variables**

To use variables effectively, we must follow Python’s naming rules:

* Variable names can only contain letters, digits and underscores (\_).
* A variable name cannot start with a digit.
* Variable names are case-sensitive (myVar and myvar are different).
* Avoid using [Python keywords](https://www.geeksforgeeks.org/python-keywords/) (e.g., if, else, for) as variable names.

## **Type Casting a Variable**

**Basic Casting Functions**

* [**int()**](https://www.geeksforgeeks.org/python-int-function/)- Converts compatible values to an integer.
* [**float()**](https://www.geeksforgeeks.org/float-in-python/)- Transforms values into floating-point numbers.
* [**str()**](https://www.geeksforgeeks.org/python-str-function/)- Converts any data type into a string.

**Casting variables**

s = "10" # Initially a string  
n = int(s) # Cast string to integer  
cnt = 5  
f = float(cnt) # Cast integer to float  
age = 25  
s2 = str(age) # Cast integer to string

## **Scope of Variable**

There are two methods how we define scope of a variable in python which are [local and global](https://www.geeksforgeeks.org/global-local-variables-python/).

Local Variable: Variables defined inside a function are local to that function.

Global Variable: Variables defined outside any function are global and can be accessed inside functions using the [global keyword](https://www.geeksforgeeks.org/global-keyword-in-python/).

## **Operators**

## **Data types**

standard or built-in data types in Python:

* **Numeric -**[int](https://www.geeksforgeeks.org/python-numbers/), [float](https://www.geeksforgeeks.org/python-float-type-and-its-methods/), [complex](https://www.geeksforgeeks.org/python-complex-function/)
* **Sequence Type -**[string](https://www.geeksforgeeks.org/python-string/), [list](https://www.geeksforgeeks.org/python-lists/), [tuple](https://www.geeksforgeeks.org/python-tuples/)
* **Mapping Type -**[dict](https://www.geeksforgeeks.org/python-dictionary/)
* **Boolean -**[bool](https://www.geeksforgeeks.org/boolean-data-type-in-python/)
* **Set Type -**[set](https://www.geeksforgeeks.org/python-sets/), [frozenset](https://www.geeksforgeeks.org/frozenset-in-python/)
* **Binary Types -**[bytes](https://www.geeksforgeeks.org/python-bytes-method/), [bytearray](https://www.geeksforgeeks.org/python-bytearray-function/), [memoryview](https://www.geeksforgeeks.org/memoryview-in-python/)

## **Conditional Statements**

Conditional statements in [Python](https://www.geeksforgeeks.org/python-programming-language-tutorial/)are used to execute certain blocks of code based on specific conditions. These statements help control the flow of a program, making it behave differently in different situations

**1. if Statement**

Executes a block of code **only if** the condition is True.

if age >= 18:

print("You are an adult.")

**2. if-else Statement**

Adds an alternative path when the condition is False.

if temperature > 30:

print("It's hot!")

else:

print("It's cool.")

**3. if-elif-else Statement**

Checks multiple conditions in sequence.

score = 85

if score >= 90:

grade = "A"

elif score >= 80:

grade = "B" *# This block executes*

else:

grade = "C"

**4. Nested Conditionals**

if statements inside other if statements.

if x > 0:

if x % 2 == 0:

print("Positive and even.")

**5. Ternary Operator (One-Liner)**

Compact if-else for simple conditions.

result = "Pass" if marks >= 40 else "Fail"

**Key Notes**

* Conditions use **comparison operators** (==, !=, >, <, etc.).
* Indentation (whitespace) defines code blocks.
* Use elif to avoid deep nesting.

## **Loops**

Loops in Python are used to repeat actions efficiently. The main types are For loops (counting through items) and While loops (based on conditions)

Python While Loop Syntax:  
**while expression:  
statement(s)**

cnt = 0 #example  
while (cnt < 3):  
 cnt = cnt + 1  
 print("Hello blah")

For Loop Syntax:  
for iterator\_var in sequence:  
statements(s)

n = 4 #example  
for i in range(0, n):  
 print(i)

**Nested Loops Syntax:**

**For loop**

for iterator\_var in sequence:  
for iterator\_var in sequence:  
statements(s)  
statements(s)

**while loop**

while expression:  
while expression:   
statement(s)  
statement(s)

## **Loop control statements**

These change execution from their normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

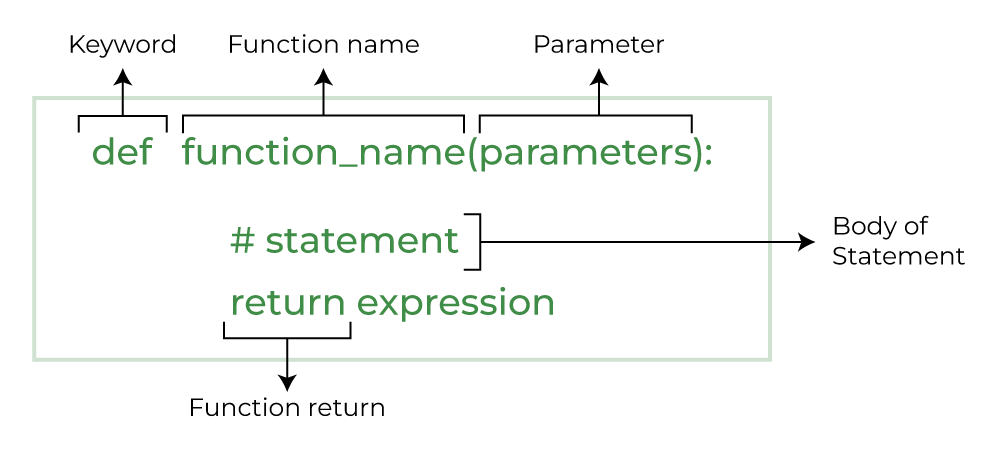
**Continue Statement**The [continue statement](https://www.geeksforgeeks.org/python-continue-statement/) in Python returns the control to the beginning of the loop and skips the current iteration

**Break Statement**The [break statement](https://www.geeksforgeeks.org/python-break-statement/) in Python brings control out of the loop and ends the iteration, i.e. no more iterations will take place

**Pass Statement**We use [pass statemen](https://www.geeksforgeeks.org/python-pass-statement/)t in Python to write empty loops. Pass is also used for empty control statements, functions and classes.

## **Functions**

**Python Functions** is a block of statements that does a specific task. The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again



## **Type of Functions**

**A. Built-in Functions**

Predefined in Python (e.g., print(), len(), input())

**B. User-defined Functions**

Created by programmers:

def greet():  
 print("Hello, World!")

# **BASICS of NumPy**

**NumPy**is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

## **1.Arrays**

an array is a structure for storing and retrieving data.

a “one-dimensional” array like a list:  
1520

A two-dimensional array would be like a table:  
1520  
8361  
1729

A three-dimensional array would be like a set of tables, perhaps stacked as though they were printed on separate pages

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index. Every item in a ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype). Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types

Most NumPy arrays have some restrictions. For instance:

* All elements of the array must be of the same type of data.
* Once created, the total size of the array can’t change.
* The shape must be “rectangular”, not “jagged”; e.g., each row of a two-dimensional array must have the same number of columns

## **2.Arrays Basics**

* Array Creation  
  Methods to create arrays:

From Python lists: np.array([1, 2, 3]).

Pre-filled arrays: np.zeros(), np.ones(), np.empty().

Ranges: np.arange(), np.linspace().

Random arrays: np.random.random().

* Array Attributes

ndarray.shape: Dimensions of the array.

ndarray.dtype: Data type of elements.

ndarray.size: Total number of elements.

* Indexing and Slicing

Similar to Python lists but extended to multi-dimensions.

Boolean and advanced indexing for filtering.

## **3.Arrays Operation**

* Mathematical Operations

Element-wise operations: +, -, \*, /, etc.

Matrix multiplication: @ or np.dot().

Aggregations: sum(), min(), max(), mean().

* Reshaping and Manipulation

Reshape: arr.reshape().

Transpose: arr.T or np.transpose().

Stacking: np.vstack(), np.hstack().

Splitting: np.hsplit(), np.vsplit().

* Broadcasting

Rules for operations on arrays of different shapes.

## **4.Array advanced topics**

* I/O with NumPy

Reading/writing arrays: np.loadtxt(), np.savetxt(), np.load(), np.save().

Handling CSV files and custom formats.

* Data Types

Supported types: int, float, bool, complex, etc.

Type conversion: arr.astype().

* Structured Arrays

Arrays with heterogeneous data types (similar to Pandas DataFrames).